Delayed Duplicate Acknowledgements: A *TCP-Unaware* Approach to Improve Performance of TCP over Wireless *

Nitin Vaidya † Miten Mehta ‡ Charles Perkins § Gabriel Montenegro ¶

Technical Report 99-003
February 1999

Abstract

Since TCP cannot distinguish between packet losses due to transmission errors from those due to congestion, TCP tends to perform poorly on wireless links that are prone to transmission errors. Several techniques have previously been proposed to improve TCP performance over wireless links. Existing schemes typically require an intermediate node (typically, a base station) to be TCP-aware. For instance, the Snoop scheme requires the base station to interpret TCP headers and take appropriate actions to help improve TCP performance. This paper proposes an alternative TCP-*unaware* technique that attempts to mimic the behavior of the Snoop protocol. Performance evaluation shows that the proposed Delayed Dupacks scheme performs quite well.

1 Introduction

Wireless networking has gained widespread acceptance in recent years [11, 12, 23, 24]. The increasing use of wireless communication makes it interesting to consider issues related to the performance of transport protocols over wireless links. In particular, this paper considers performance of TCP, the Transmission Control Protocol for connection-oriented reliable in-order data delivery [22], commonly used in the present-day Internet.

Wireless links are often prone to a higher transmission error rate, when compared to wired links. Because of its inability to distinguish between packet losses due to congestion and due to transmission errors, TCP performs poorly on wireless links [2, 3]. Several interesting approaches have already been proposed to improve TCP performance over wireless [2, 4, 3, 5, 8, 10, 15, 17, 19, 21, 25, 26, 27]. However, many of these schemes require TCP-specific actions on the part of intermediate nodes on the path from TCP sender to TCP receiver. Such schemes are sometimes referred as being *TCP-aware* [3].

The objective of this paper is to investigate an alternative "TCP-unaware" technique that can improve TCP performance *without* taking TCP-specific actions at intermediate nodes. In particular, the TCP-*unaware* scheme proposed here attempts to *imitate* a previously proposed *TCP-aware* scheme named Snoop [4].

*This research is supported in part by the Texas Advanced Technology Program grant 010115-248, National Science Foundation grant 32525-46600, and Sun Microsystems.

†Nitin Vaidya is the contact author. He can be reached at: Department of Computer Science, Texas A&M University, College Station, TX 77843-3112, E-mail: vaidya@cs.tamu.edu, Phone: 409-845-0512, Fax: 409-847-8578

‡Microsoft Corporation, Redmond, WA. Mehta performed a part of this work at the Texas A&M University.

§Sun Microsystems Laboratories, Palo Alto

¶Sun Microsystems Laboratories, Palo Alto

2 Performance of TCP over Wireless Links

We begin with a brief summary of relevant TCP features.

TCP Overview [22]: TCP achieves reliability by requiring that the TCP sender retransmit lost packets. For this purpose, the TCP receiver acknowledges receipt of data packets from the sender. An acknowledgement (or `ack`) sent by the TCP receiver is *cumulative*. A new packet received by the receiver is said to be out-of-order (OOO) if it is not the next in-sequence packet expected by the receiver. On receiving this out-of-order packet, the receiver sends a *duplicate acknowledgement* (or `dupack`), acknowledging all the in-sequence bytes received so far. The TCP sender determines that a packet is lost using one of the following two mechanisms [22]:

- Retransmission timeout: If a timer, set when a packet is transmitted, expires before acknowledgement of the packet is received, the packet is presumed lost.
- Fast retransmit: If three dupacks containing sequence number P are received, then the packet that includes sequence number P is presumed lost, and retransmitted. For fast retransmission to occur, the receiver must receive at least three out-of-order packets consecutively.

The TCP sender maintains a *congestion window* that determines the maximum amount of unacknowledged data sent by the sender. When a packet loss is detected, the TCP congestion control mechanism drastically reduces the congestion window size, effectively reducing the amount of data sent by the sender in one round-trip time (RTT).

Performance of TCP Over Wireless: TCP can perform poorly over wireless links [2, 3, 13] due to the assumption that primary cause of packet losses is congestion. On wireless links, packet losses due to transmission errors may be non-negligible. However, the TCP sender, on detecting such a packet loss, reduces its congestion window, unnecessarily degrading throughput [2, 3].

In wireless links, using link level retransmission to remedy transmission errors may result in adverse interaction with TCP:

- If the wireless link level retransmissions take too long, the TCP sender may time out before the retransmitted packet could be delivered to, and acknowledged by, the TCP receiver [13].
- If the wireless link level retransmission scheme delivers packets out-of-order (OOO), then the OOO packets would result in duplicate acknowledgements from the receiver. If the TCP sender receives three duplicate acknowledgements, the fast retransmit mechanism will be triggered [4].

In each of the above cases, the TCP sender would retransmit the lost packet, leading to the so-called "interference" between TCP and link level retransmissions. This interference wastes wireless bandwidth by duplicating retransmissions [13]. In addition, since the TCP sender detects that a packet is lost, it reduces its congestion window.

3 Related Work

Several approaches to improve performance of TCP over wireless links have been proposed. In the split connection approach, a TCP connection is broken into two TCP connections – one from the sender to the base station, and another from the base station to the receiver [2, 26]. Thus, wireless errors can be handled *locally*, by means of retransmissions from the base station. This approach, however, violates the end-to-end reliability semantics of TCP.

The Snoop protocol [4], described later in the paper, also performs local recovery, but improves on the split connection approach by retaining the end-to-end reliability semantics. WTCP [21] modifies the Snoop protocol by time-stamping packets. The time-stamps are used to provide the sender with a more accurate estimate of round-trip times, despite retransmissions on the wireless link.

The explicit loss notification (ELN) scheme [3] assumes that the receiver can determine the cause of a packet loss, and send a notification to the sender. Other explicit notification schemes have also been proposed [14, 7, 10].

Although earlier research on TCP over wireless often focussed on ways to modify TCP, some recent research has focussed on tuning the link layer implementation to allow TCP to achieve better throughput [20, 15, 19, 18, 9].

Host mobility is an issue related to the problem considered here, since wireless hosts are often mobile. To our knowledge, Caceres et al. [6] were the first to consider the impact of mobility on TCP performance, and suggest a mechanism to improve the throughput. Several other researchers have also considered techniques designed to take mobility into account [2, 6, 5, 17, 25].

4 The Snoop Scheme

Because the proposed TCP-unaware scheme attempts to imitate the TCP-aware Snoop scheme [4], we first describe Snoop briefly. For the discussion below, consider the system shown in Figure 1. Consider TCP data transfer from node S to the wireless host WH, through the base station BS. The link between S and BS is wired, whereas the link between BS and WH is wireless.

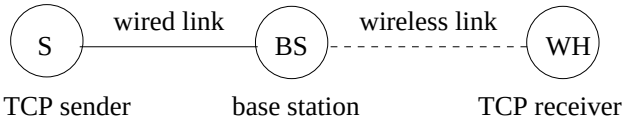


Figure 1: System model

The essential elements of the Snoop scheme [4] may be summarized as follows:

- Link level retransmissions: Snoop uses a link level retransmission mechanism that allows the *base station* to retransmit packets lost due to transmission errors on the wireless link. The base station BS caches TCP packets that have been sent to the wireless host WH, but for which a TCP acknowledgement has not been received from WH. When the base station receives a duplicate acknowledgement with sequence number P from WH, the base station retransmits the packet containing sequence number P, provided that the packet is cached at

the base station. (Since a single packet loss due to errors may results in multiple duplicate acknowledgements, Snoop takes care to avoid retransmitting the packets unnecessarily.)

The link level retransmission scheme in Snoop may deliver packets out-of-order over the wireless link. As discussed earlier, link level retransmissions may interfere with retransmissions by the TCP sender (which may be caused either by a retransmission timeout or fast retransmit). Snoop is designed for environments where the likelihood of a timeout at the TCP sender is small (this is often true, since TCP retransmission timeouts typically use coarse granularity [4]). However, out-of-order packet delivery over the wireless link can trigger a fast retransmit from the TCP sender. The second feature of Snoop described below avoids this.

- Reducing interference between TCP retransmissions and link level retransmissions: As noted above, retransmissions from the base station are triggered by the receipt of duplicate acknowledgements from WH. If these duplicate acknowledgements are forwarded to the sender node S, then the TCP sender would *fast retransmit* on receipt of three duplicate acknowledgements. To avoid fast retransmit by the sender, the base station *drops* duplicate acknowledgements with sequence number P if the base station can locally retransmit the packet containing sequence number P. Now, the TCP sender will not fast retransmit, since it will not receive the duplicate acknowledgements.

Snoop uses some other features (such as a link level retransmission timeout) as well, however, we limit the discussion above to the most important features of Snoop.

Figure 2 illustrates operation of the Snoop scheme with an example (please see caption of the figure for a discussion).

Because the scheme proposed in this paper is designed to behave similar to Snoop, the proposed scheme would tend to perform poorly whenever Snoop performs poorly. In particular, on slow wireless links, the round-trip time on the wireless link may be large (large on an absolute scale, and also a large fraction of the end-to-end round-trip time of the TCP connection). In such cases, despite large granularity of the TCP retransmission timer, the TCP sender is likely to timeout while a retransmission is being performed on the wireless link. Therefore, protocols such as Snoop (and the proposed scheme) cannot perform well in these environments, and alternative techniques need to be used. Similar to [4, 3], in our simulation study, we choose the wireless bandwidth such that a wireless link level retransmission can potentially be performed before the TCP sender may timeout. This assumption is valid, for instance, when a wireless LAN (such as AT&T WaveLAN) is used.

5 Proposed Delayed Dupacks Scheme

In the proposed TCP-unaware scheme, the base station does not need to look at the TCP headers. The proposed scheme may be preferred over Snoop when encryption is used. To provide security, packets may be encrypted by the sender, as specified by IPSEC [1] or some other standard. With encryption, the base station may not be able to view the TCP headers. In such cases, the Snoop scheme is not useful.

Proposed Delayed Duplicate Acknowledgements (or, Delayed Dupacks) scheme attempts to approximate the behavior of Snoop. The proposed scheme can be summarized as follows:

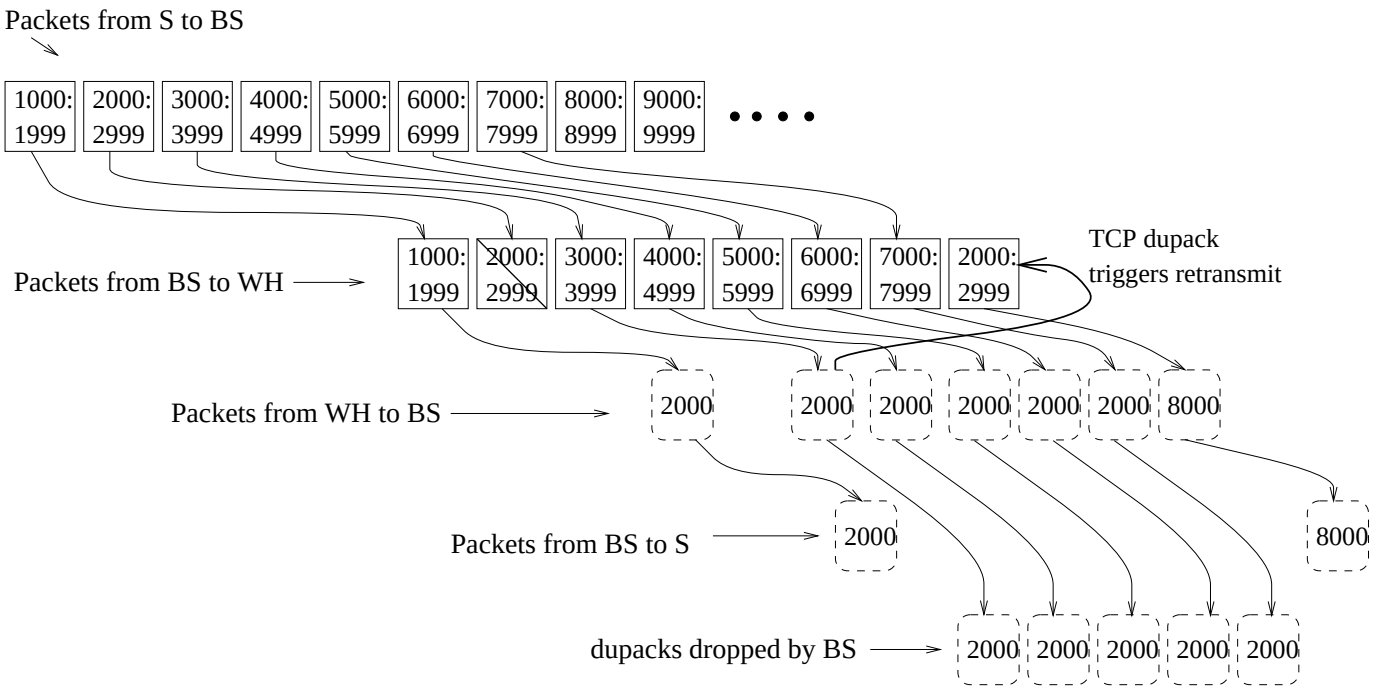


Figure 2: *An illustration of the Snoop scheme: In this figure, time proceeds from left to right. Although the figure is not drawn to scale, various events are shown in the correct order. The boxes containing two sequence numbers denote TCP data packets, whereas the boxes containing a single sequence number are TCP acks. For instance, the box containing 2000:2999 denotes a TCP packet that contains the 1000 bytes with sequence numbers 2000 through 2999. Recall that a TCP acknowledgement that contains sequence number, say 2000, denotes that the receiver has received all bytes through 1999, but not byte 2000. Diagonal line through the data packet 2000:2999 sent by BS denotes that the packet is lost due to transmission errors. An arrow from packet X to packet Y denotes that packet X is the cause for packet Y. Observe that the base station retransmits packet 2000:2999 on receipt of the first dupack. Also, the base station drops the dupacks with sequence number 2000. The TCP sender does not receive any of these dupacks, and remains unaware of the transmission error.*

- Link level retransmissions: The base station implements a link level retransmission scheme for packets that are lost on the wireless link due to transmission errors. As with Snoop, the link level retransmission scheme does *not* attempt to deliver packets in-order. While Snoop uses TCP duplicate acknowledgements (dupacks) to trigger link level retransmissions, our scheme uses *link level* acknowledgements. Therefore, the proposed scheme can be implemented without making the base station TCP-aware.

- Reducing interference between TCP retransmissions and link level retransmissions: In Snoop, the base station drops duplicate acknowledgements to avoid unnecessary fast retransmits from the TCP sender. Instead, in the Delayed Dupacks scheme, the TCP receiver attempts to reduce interference between TCP and link-level retransmissions by *delaying* third and subsequent dupacks for some interval d. Specifically, when out-of-order (OOO) packets are received, the TCP receiver responds to the first two consecutive OOO packets by sending

dupacks immediately. However, dupacks for further consecutive OOO packets are delayed for duration d. If the next in-sequence packet is received within the d interval, then the delayed dupacks are not sent. Otherwise, after the d interval, all the delayed dupacks are released.

Now consider two types of packet losses that may occur.

- Case 1 – A packet is lost due to transmission errors on the wireless link: If d is chosen large enough to allow time for link level retransmission (from the base station) of the lost packet, then the retransmitted packet would reach the TCP receiver before third and subsequent dupacks could be sent. In this case, the delayed dupacks will not be sent, and on receiving the retransmitted packet, the receiver will acknowledge receipt of all the in-sequence data (including the retransmitted packet). Since the TCP sender does not receive more than two duplicate acknowledgements, it will not fast retransmit. Thus, in the case of packet losses due to transmission errors, the Delayed Dupacks scheme can imitate Snoop, with the appropriate choice of dupack delay d.

- Case 2 – A packet is lost due to congestion before reaching the wireless link: In this case, the wireless link level retransmission mechanism is not useful. However, the TCP receiver will delay the third and subsequent dupacks for d time units (if three or more OOO packets are received). This can lead to a degradation in performance, compared to standard TCP. Standard TCP will send the third dupack without any delay, thus initiating fast retransmission sooner than the proposed protocol.

  Also, after the d interval, the TCP receiver could send multiple dupacks in a burst. This is not a serious issue when the delay-bandwidth product on the wireless link is not very large (in this case, the burst of dupacks will be small). However, if the burst is large, some mechanism to pace the dupacks may need to be used.

The above discussion suggests that the proposed approach may be beneficial when packet losses due to wireless transmission errors occur, but could be detrimental when packet losses due to congestion occur.[1] The overall performance improvement or degradation depends on the relative frequency of the two types of packet losses. As seen later in the paper, the Delayed Dupacks scheme can significantly improve performance when transmission errors occur, without degrading the performance much when transmission errors do not occur.

The Link Level Retransmission Scheme

The proposed scheme relies on a link level retransmission scheme to retransmit packets that are lost due to wireless errors. As such, any link level retransmission scheme could potentially be used in conjunction with the proposed Delayed Dupacks scheme (recall that the Delayed Dupacks scheme only modifies the behavior of the TCP receiver).

For the purpose of simulation experiments, we simulated a simple scheme that sends a link level acknowledgement for each link level data packet. Each TCP packet (i.e., TCP data or TCP acknowledgement) is encapsulated in one link layer data packet. Receipt of a link level data packet is acknowledged by a link level ack that contains sequence number of the received link layer packet. The link layer uses its own sequence numbers space (independent of TCP sequence numbers). Loss

[1] We will use the term *congestion loss* to refer to a packet loss that occurs due to congestion.

of link layer packet X is detected when a link layer ack for a link layer packet Y, such that $Y > X$, is received. On detecting the loss of a link layer data packet, the lost packet is retransmitted on the wireless link. Similar to Snoop, this scheme also uses a retransmission timer at the link layer, although typically link layer retransmission is triggered by the link layer acks, not by the timer.

The link layer gives higher priority to link layer acks, as compared to link layer data. Similarly, retransmitted link layer data packets are given higher priority compared to other link layer data packets. This priority mechanism is used to speed up detection and recovery of packet losses due to transmission errors.

The link layer scheme used here can potentially be improved, for instance, by piggybacking link level acks onto link level data. However, we show that despite the use of a simple link level retransmission scheme, the Delayed Dupacks scheme can perform quite well.

An Example

Figure 3 illustrates the delayed dupacks scheme with an example. A comparison of this figure with Figure 2 shows the similarities as well as differences in the behavior of Snoop and Delayed Dupacks schemes.

6 Simulation Results

We evaluated performance of the Delayed Dupacks scheme using the *ns-2* simulator [16]. Figure 1 illustrates the network topology used in our evaluation. A one-way TCP transfer is established from node S to node WH. The link between the sender node S and base station BS is wired and full-duplex. The wired link bandwidth and delay are fixed at 10 Mbps and 20 ms, respectively. The link between the base station BS and the wireless host WH is wireless and full-duplex. The wireless link bandwidth is fixed at 2 Mbps. We considered four values of wireless link delay (1 ms, 10 ms, 20 ms and 40 ms) in our simulations. Only results for wireless link delay of 1 ms and 20 ms are reported here due to lack of space.

The TCP source is assumed to be performing a bulk data transfer. TCP data packets contain 1000 bytes, while TCP acks contain 40 bytes. Link level acks (used when simulating the Delayed Dupacks scheme) contain 14 bytes. For the Delayed Dupacks scheme, the dupack delay d was varied in the range 0 to 0.2 second.

In the ns-2 simulation model, a queue is associated with each direction of a link. In our simulations, each queue can hold at most 40 packets (queue size is measured as the *number* of packets in the queue). Recall that the link level retransmission scheme used with Delayed Dupacks sends link level acks, while Snoop uses TCP acknowledgements instead. Therefore, if a single queue is used for all packets on the wireless link, the queue would overflow much sooner when using the Delayed Dupacks scheme as compared to Snoop. To avoid this unfairness to the Delayed Dupacks scheme, on the wireless link, we use two queues: one queue for link layer data and another queue for link layer acks. Note that TCP data and TCP acks are both considered to be link layer *data*. The additional queue for link layer acks does not increase the memory requirement much (40 link layer acks need much less than 1000 bytes).

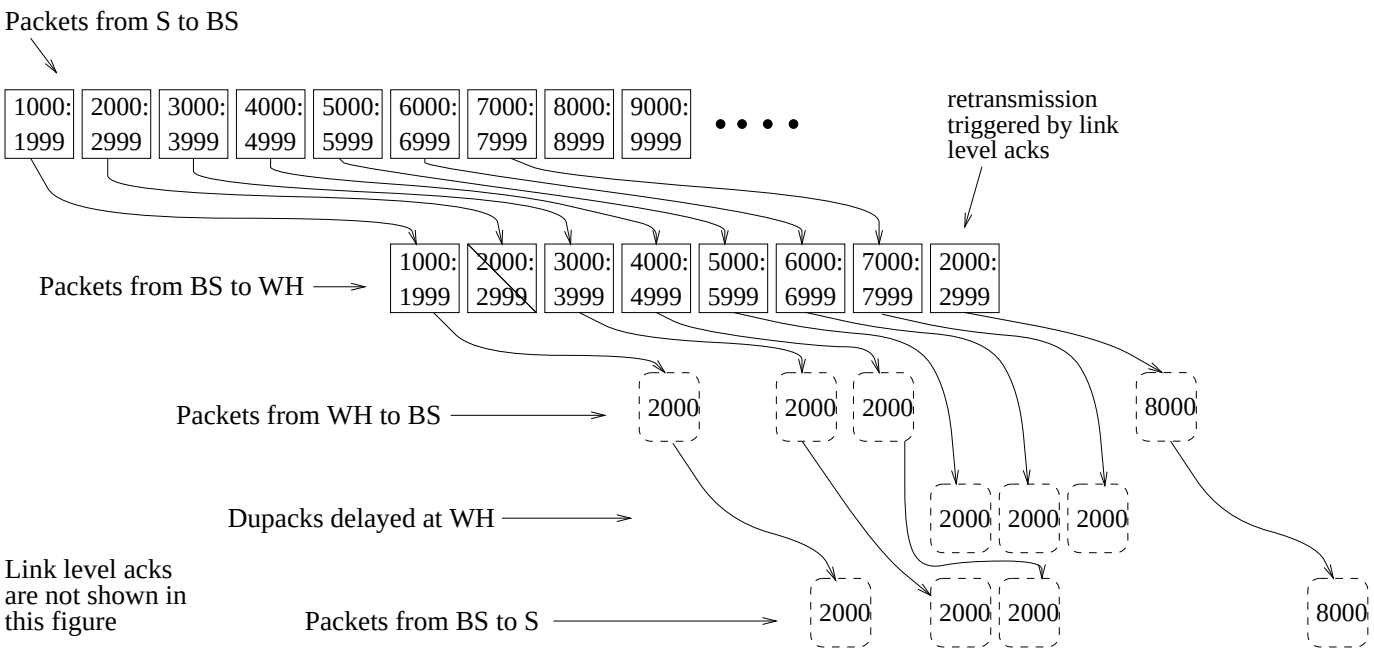


Figure 3: *An illustration of the Delayed Dupacks scheme: This figure uses notation similar to Figure 2. Although the Delayed Dupacks scheme uses link level acknowledgement, they are not shown in the figure. Unlike Snoop, in this case, the retransmission of packet 2000:2999 by the base station is triggered by link level acks (not shown in the figure). The first two dupacks with sequence number 2000 are sent by the TCP receiver to the base station, and propagated by the base station to the TCP sender. The third and subsequent dupacks are delayed at the receiver. In this case, the retransmitted packet is assumed to be delivered to the receiver within the d interval. Hence, receiver WH will discard the delayed dupacks when the retransmitted packet is received.*

Transmission errors on wireless link can occur in both directions. The error model is exponential. For the simulations, the mean number of bytes between errors (or 1/Error Rate) was chosen to be 16384, 32768, 65536 or 131072 bytes. Erroneous packets are dropped on receipt.

We simulate congestion elsewhere in the network by dropping TCP data packets on the wired links (when sent from TCP sender to BS). The *uniform* error model from ns-2 [16] is used for this purpose. Congestion loss rates of 0.0, 0.01, 0.03 and 0.05 per packet are used in the simulations.

All TCP transfers begin somewhere between 0 and 1 seconds from start of the simulation. For congestion loss rate of 0 and 0.01 per packet, simulation terminates at 20 seconds, while for congestion loss rate of 0.03 and 0.05 per packet, the simulation terminates at 60 and 80 seconds, respectively. Each measurement reported in the graphs in this section is averaged over 12 simulation runs.

For the simulations, "base TCP" is implemented using ns-2's TCP/Reno agent for the sender, and TCPSink agent for the receiver [16]. The Delayed Dupacks scheme is implemented by modifying the TCPSink agent, and by adding a simulation module for the link level retransmission scheme.

As discussed above, we vary four parameters in our simulations: (1) wireless transmission error rate, (2) wireless link delay, (3) congestion loss rate, and (4) delay interval d for the Delayed Dupacks scheme. In addition, we also performed measurements varying the number of TCP connections sharing the wireless link from 1 to 4. However, due to lack of space, we only report measurements for a single TCP connection. We compare four schemes:

- Base TCP (without any link level retransmission mechanism): As noted above, base TCP is simulated using the TCP/Reno and TCPSink agents in the ns-2 simulation package.
- Base TCP on top of the (TCP-unaware) link level retransmission scheme:

  Note that when the Delayed Dupacks scheme is used with $d = 0$, behavior of the TCP receiver becomes identical to that in the base TCP. Therefore, all results presented below for $d = 0$ should be considered to be for base TCP executed on top of our TCP-unaware link level retransmission scheme.
- Delayed Dupacks scheme.
- Snoop scheme – note that when simulating the Snoop scheme, our link layer protocol is not used.

We now present the simulation results.

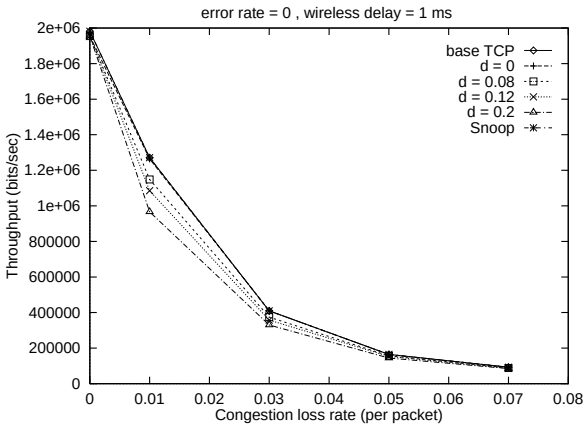
6.1 Results with No Transmission Errors

As noted previously, the Delayed Dupacks scheme may delay recovery from congestion-related packet losses. This can have a detrimental impact on TCP performance. To estimate the detrimental impact in the worst case, we first consider the case when no transmission errors occur on the wireless link. Under this condition, we would like the Delayed Dupacks scheme to perform close to base TCP.

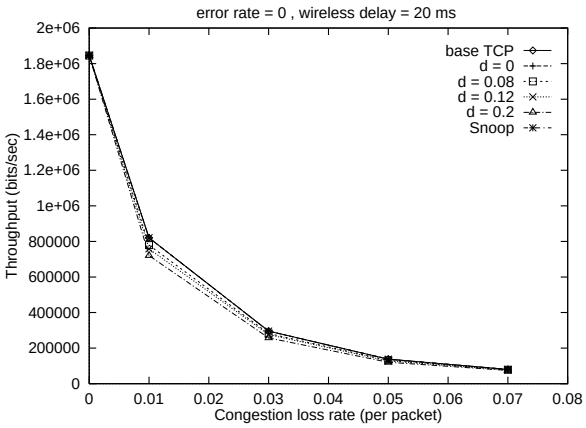
Figure 4 plots TCP throughput versus congestion loss rate, for wireless delay of 1 ms and 20 ms, assuming that transmission error rate is 0. In the figure, the curves for $d = 0$, $d = 0.08$, etc., correspond to the Delayed Dupacks scheme with the corresponding value of dupack delay d. Note that the delay listed in the figures is in seconds – thus, the curve for $d = 0.08$ corresponds to dupack delay of 0.08 second or 80 ms. Some observations based on Figure 4 are as follows:

- As should be expected, when congestion loss rate is increased, throughput decreases.

  The curves for wireless link delay of 1 ms and 20 ms show similar trends. The throughput for the case of 20 ms wireless link delay is lower than that for 1 ms, since larger end-to-end round-trip time results in slower congestion window growth for a TCP connection.
- Throughput for Snoop is similar to base TCP – the curves for base TCP and Snoop overlap in the figure. When packets are lost only due to congestion, Snoop behaves similar to base TCP, yielding similar performance.
- Delayed Dupacks uses the wireless channel for *link level* acks, unlike base TCP or Snoop. Therefore, Delayed Dupacks yields lower throughput than base TCP even for congestion loss rate of 0.



(a) Wireless delay = 1 ms



(b) Wireless delay = 20 ms

Figure 4: No transmission errors: Throughput (bits/sec) versus congestion loss rate (per packet)

For non-zero congestion loss rate, throughput of the Delayed Dupacks scheme monotonically decreases when d is increased. Since the transmission error rate is 0, all packet losses are due to congestion. In this case, delaying dupacks only delays recovery from congestion losses, thus degrading throughput. Therefore, a larger dupack delay results in worse throughput.

- The throughput degradation for Delayed Dupacks scheme is worst for moderate values of congestion loss rate. When congestion loss rate is small, there are fewer opportunities for unnecessarily delaying dupacks at the TCP receiver, therefore, the performance degradation is small. When congestion loss rate is large, retransmission timeouts at the TCP sender become frequent – the degradation due to these timeouts becomes more prominent than that due to the delayed dupacks. Timeouts affect performance of all schemes (including base TCP). Therefore, the relative degradation due to delayed dupacks is small when the congestion loss rate is high.

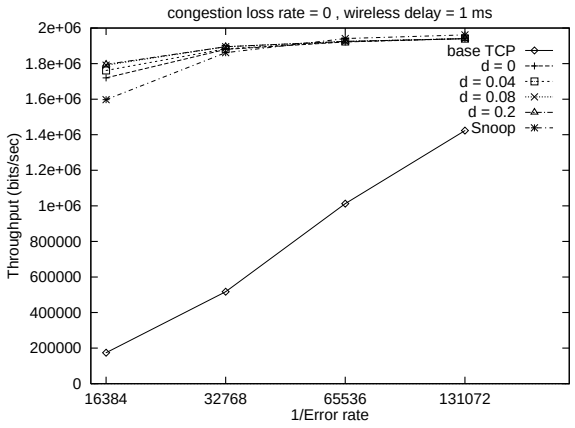
Results for wireless link delays of 10 ms and 40 ms are similar to the above results, and omitted due to lack of space.

In the rest of the paper, we consider the case when transmission errors may occur.

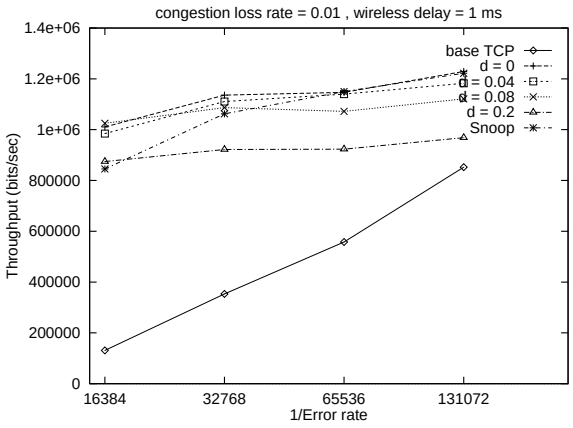
6.2 With Transmission Errors: Wireless Link Delay 1 ms

Figure 5 plots TCP throughput versus 1/Error Rate when wireless link delay is 1 ms (note that the results reported in the figures are averages over 12 runs). The four sets of curves in Figure 5 correspond to four different congestion loss rates. In Figure 5, base TCP (without link level retransmissions) performs poorly, as expected. However, base TCP on top of the TCP-unaware link layer retransmission scheme (i.e., Delayed Dupacks scheme with $d = 0$) performs quite well. In other words, Delayed Dupacks and Snoop are *unable* to achieve performance much *better* than simply using standard TCP in conjunction with a reliable link layer mechanism.

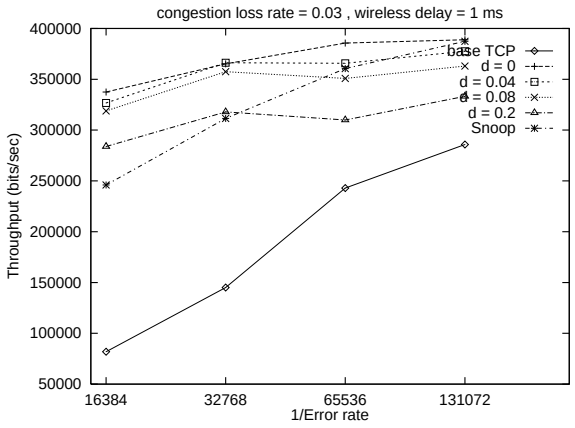
The explanation for this observation is quite simple. As discussed earlier, Delayed Dupacks and Snoop both attempt to achieve better performance by avoiding TCP fast retransmit when



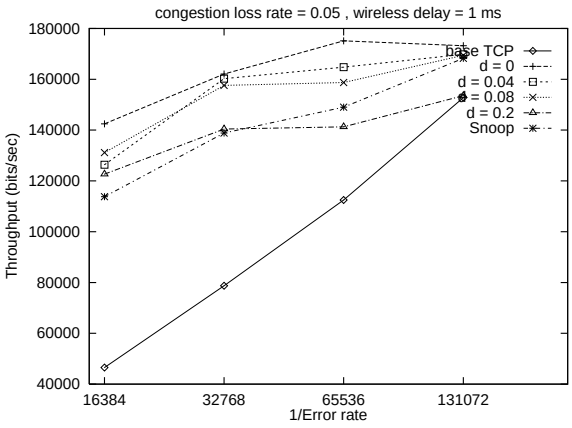
(a) Congestion loss rate = 0.0 per packet



(b) Congestion loss rate = 0.01 per packet



(c) Congestion loss rate = 0.03 per packet



(d) Congestion loss rate = 0.05 per packet

Figure 5: Wireless link delay = 1 ms

packets are lost due to wireless transmission errors, despite out-of-order delivery at the link layer. Consider placement of a packet retransmitted by the wireless link layer. Figure 6 shows two possibilities. In this figure, a diagonal line through a packet denotes that it was lost due to transmission errors. The packets are sent by the base station BS to receiver WH in the left-to-right order in each case.

- Case A: In this case, packet 1 is lost due to transmission errors, and the base station sends packets 2 and 3 before it retransmits packet 1. Thus, the number of out-of-order (OOO) packets received by the receiver, before it receives the retransmitted packets, is less than three.
- Case B: In this case also, packet 1 is lost due to transmission errors, but the base station sends packets 2 through 5 before retransmitting packet 1.

In case A, even if standard TCP is used, the packet loss due to transmission errors will not cause fast retransmit at the sender (since three OOO packets are not delivered to the receiver). Thus, fast retransmit is likely to occur only in cases similar to Case B. To put it differently, fast retransmit is

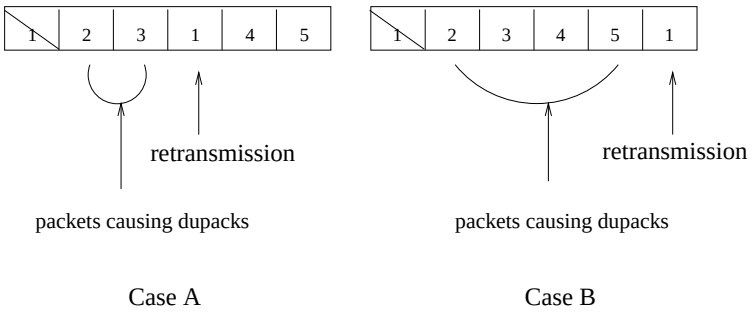


Figure 6: Placement of retransmitted packets

likely to occur only when the delay-bandwidth product of the wireless link is at least 4 data packets. For Figure 5, the one-way wireless link delay is 1 ms and wireless bandwidth is 2 Mbps. Since the delay-bandwidth product is quite small, Snoop or Delayed Dupacks (with $d > 0$) cannot yield performance improvement compared to executing base TCP in conjunction with the TCP-unaware link level retransmission scheme (i.e., Delayed Dupacks with $d = 0$).

Our measurements indicate that, when the error rate is high, Snoop often performs somewhat worse than Delayed Dupacks. This occurs because the link layer retransmission scheme is able to recover from multiple packet losses better than Snoop. As noted in [3], in such situations, Snoop performance can be improved by using selective acknowledgements (SACK).

In Figures 5(a) through 5(d), observe that as the congestion loss rate increases, the difference between throughput for the best and worst values of d plotted increases. This occurs due to the delayed reaction by the Delayed Dupacks scheme when congestion losses occur.

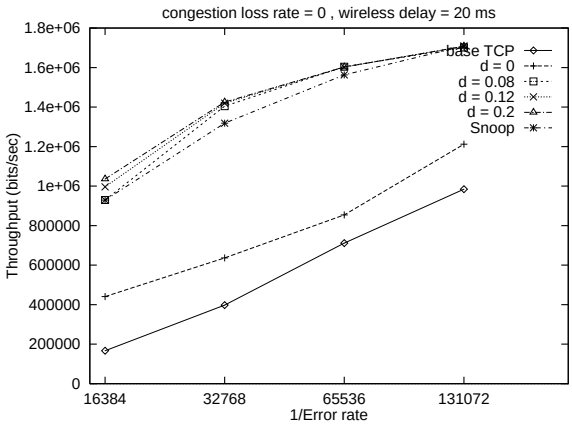
We observed that for large values of delay d, sometimes when the error rate is decreased, the average throughput decreases slightly. For instance, this occurs for congestion loss rate of 0.03 and $d = 0.2$ second. We have not yet conclusively explained this phenomenon (statistical variations is a plausible cause). Further simulation experiments are being performed to determine the cause.

6.3 With Transmission Errors: Wireless Link Delay 20 ms

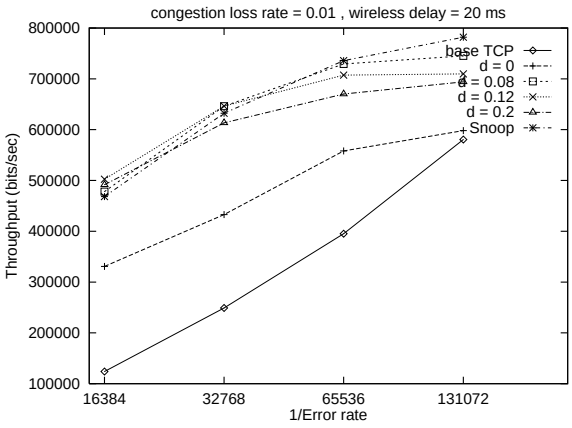
Unlike the case with wireless link delay of 1 ms, when the wireless link delay is 20 ms, the delay-bandwidth product for the wireless link is larger than 4 packets (recall that in our simulations TCP data packet size is 1000 bytes) Therefore, earlier discussion of Figure 6 suggests that Snoop and Delayed Dupacks (with suitable d) should be able to perform better than base TCP. For wireless delay of 20 ms, Figure 7 plots TCP throughput versus 1/Error Rate. The four sets of curves in Figure 7 correspond to four different congestion loss rates.

In Figure 7(a), with congestion loss rate = 0, Delayed Dupacks scheme with $d = 0$ performs much worse than Snoop and Delayed Dupacks with larger d. Since the wireless delay is 20 ms, the round-trip delay on the wireless link is greater than 40 ms. Thus, if d is small (say, smaller than 40 ms), the delayed dupacks are released before the link layer could deliver the retransmitted packet. Therefore, with small d, fast retransmit at the sender cannot be prevented, resulting in poor throughput. However, the throughput for $d = 0$ is marginally better than base TCP, since the link layer recovers from the packet losses sooner than the base TCP.

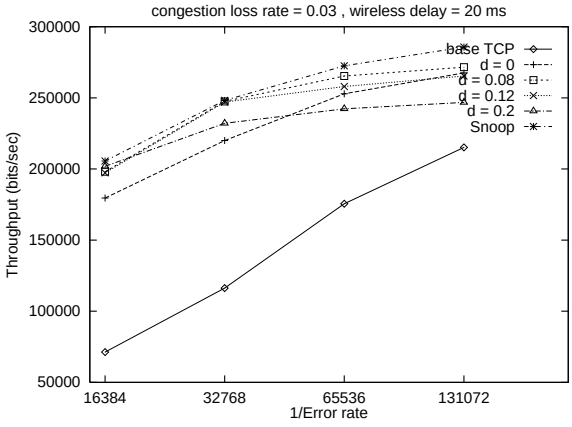
Again, in Figure 7(a), with congestion loss rate = 0, observe that the throughput of the



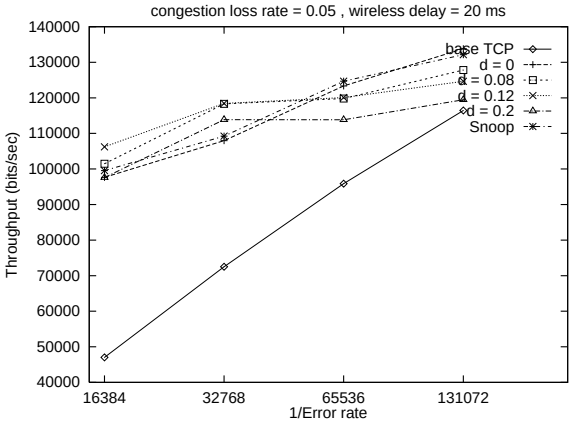
(a) Congestion loss rate = 0.0 per packet



(b) Congestion loss rate = 0.01 per packet



(c) Congestion loss rate = 0.03 per packet



(d) Congestion loss rate = 0.05 per packet

Figure 7: Wireless link delay = 20 ms

Delayed Dupacks scheme, is comparable with Snoop when an appropriately large delay d is chosen. Since, in this case, congestion loss rate is 0, all packet losses are due to transmission errors. Therefore, larger d has the beneficial effect of avoiding fast retransmits even if multiple retransmissions of a packet are needed on the wireless link (particularly, at high error rates). Therefore, in the absence of congestion losses, larger d tends to perform better than small d. This phenomenon is the reverse of that observed in Figure 4.

In Figures 7(a) through 7(d), note that the Delayed Dupacks scheme with $d = 0.8$ performs quite well, and approaches the performance of Snoop. Since the round-trip delay on the wireless link is approximately 40 ms, $d = 80$ ms is sufficient to allow for one retransmission of a lost packet. Also, the probability that multiple retransmissions would be needed to recover from a transmission error is small. Thus, the delay of $d = 80$ ms is typically sufficient to successfully retransmit the lost packet. Since the delay of 80 ms is not too large compared to end-to-end round-trip time for the TCP connection, the impact of delaying fast retransmit in the event of congestion losses is not too detrimental. Thus, $d = 80$ ms performs well for all the congestion loss rates simulated in our measurements.

Due to lack of space, we omit the simulation results when wireless link delay is 10 ms or 40

ms. The results for these cases also show trends similar to the results presented above.

The simulation results presented above show that when the delay-bandwidth product on the wireless link is small, it is not *necessary* to use the Delayed Dupacks scheme or the Snoop scheme to achieve good performance. Standard TCP executed on top of a link level retransmission scheme should suffice in such cases. However, when the delay-bandwidth product of the wireless link is somewhat larger, the Delayed Dupacks scheme can be beneficial. The performance of the Delayed Dupacks scheme is comparable with Snoop, for appropriately chosen value of dupack delay d. Also, it is possible to achieve good performance using the same value of d over a reasonable range of congestion loss rates (we simulated 0 to 5% congestion loss rates).

7 Conclusions and Further Work

Snoop [4] uses a TCP-aware link layer scheme to improve performance of TCP over wireless links prone to transmission errors. This paper presents a TCP-unaware scheme, named Delayed Duplicate Acknowledgements, that attempts to imitate the behavior of Snoop. Performance measurements show that the Delayed Dupacks scheme can often perform comparable to Snoop, by using the *appropriate* value for the dupack delay d. Additional work is needed to address the following issues:

- How to determine the optimal value of delay d? It would be useful to derive some guidelines for choosing appropriate d.
- What is the impact of shared wireless media on the suitable value of d? When TCP connections to several destination hosts share the wireless medium, the appropriate value of d could be affected by the number of hosts sharing the medium.
- How does the end-to-end round-trip time (RTT) affect performance? Additional measurements are needed to evaluate how the relationship between end-to-end RTT and RTT on the wireless link affects performance of the Delayed Dupacks scheme.

References

[1] R. Atkinson, "Security architecture for the internet protocol," Tech. Rep. RFC 2401, IPSEC working group, Internet Engineering Task Force, November 1998.

[2] A. Bakre and B. Badrinath, "I-TCP: Indirect TCP for mobile hosts," in *Proc. 15th International Conf. on Distributed Computing Systems (ICDCS)*, May 1995.

[3] H. Balakrishnan, V. Padmanabhan, S. Seshan, and R. Katz, "A comparison of mechanisms for improving TCP performance over wireless links," in *ACM SIGCOMM, Stanford, CA*, August 1996.

[4] H. Balakrishnan, S. Seshan, and R. Katz, "Improving reliable transport and handoff performance in cellular wireless networks," *ACM Wireless Networks*, vol. 1, December 1995.

[5] K. Brown and S. Singh, "M-TCP: TCP for mobile cellular networks," *ACM Computer Communications Review*, vol. 27, no. 5, 1997.

[6] R. Caceres and L. Iftode, "Improving the performance of reliable transport protocols in mobile computing environments," *IEEE Journal on Selected Areas in Communications*, vol. 13, June 1995.

[7] K. Chandran, S. Raghunathan, S. Venkatesan, and R. Prakash, "A feedback based scheme for improving TCP performance in ad-hoc wireless networks," tech. rep., Computer Science, University of Texas-Dallas, October 1997.

[8] H. Chaskar, T. V. Lakshman, and U. Madhow, "On the design of interfaces for TCP/IP over wireless," in *MILCOM*, 1996.

[9] A. Chockalingam, M. Zorzi, and R. R. Rao, "Performance of TCP on wireless fading links with memory," in *Proc. of IEEE ICC '98*, June 1998.

[10] J. A. Cobb and P. Agrawal, "Congestion or corruption? a strategy for efficient wireless TCP sessions," in *IEEE Symposium on Computers and Communications, Alexandria, Egypt*, pp. 262–268, 1995.

[11] D. C. Cox, "Wireless personal communication: What is it?," *IEEE Personal Communication*, pp. 20–35, April 1995.

[12] R. A. Dayem, *Mobile Data and Wireless LAN Technologies*. Prentice Hall, 1997.

[13] A. DeSimone, M. Chuah, and O. Yue, "Throughput performance of transport-layer protocols over wireless LANs," in *Proc. Globecom '93*, December 1993.

[14] R. Durst, G. Miller, and E. Travis, "TCP extensions for space communications," in *Proceedings of MOBICOM '96*, pp. 15–26, November 1996.

[15] D. A. Eckhardt and P. Steenkiste, "Improving wireless LAN performance via adaptive local error control," in *Int. Conf. Network Protocols*, pp. 327–338, 1998.

[16] K. Fall and K. Varadhan, "*ns* Notes and documentation," tech. rep., VINT Project, UC-Berkeley and LBNL, 1997.

[17] Z. Haas and P. Agrawal, "Mobile-TCP: An asymmetric transport protocol design for mobile systems," in *ICC'97, Montreal, Canada*, June 1997.

[18] P. Lettieri, C. Schurgers, and M. B. Srivastava, "Adaptive link layer strategies for energy efficient networking." submitted for publication.

[19] P. Lettieri and M. B. Srivastava, "Adaptive frame length control for improving wireless link throughput, range, and efficiency," in *INFOCOM*, pp. 564–565, 1998.

[20] U. Madhow, "Dynamic congestion control and error recovery over a heterogeneous internet (invited paper)," in *IEEE CDC*, 1997.

[21] K. Ratnam and I. Matta, "WTCP: An efficient transmission control protocol for networks with wireless links," Tech. Rep. NU-CCS-97-11, Northeastern University, July 1997.

[22] W. R. Stevens, *TCP/IP Illustrated*. Addison-Wesley, 1994.

[23] Telxon Corporation, "Wireless LAN products," November 1998.

[24] B. Tuch, "Development of WaveLAN, an ISM band wireless LAN," *AT&T Technical Journal*, pp. 27–37, July/August 1993.

[25] K. Wang and S. K. Tripathi, "Mobile-end transport protocol: An alternative to TCP/IP over wireless links," in *IEEE Infocom*, pp. 1046–1053, March 1998.

[26] R. Yavatkar and N. Bhagwat, "Improving end-to-end performance of TCP over mobile internetworks," in *Workshop on Mobile Computing Systems and Applications*, December 1994.

[27] M. Zorzi and R. R. Rao, "Error control in multi-layered stacks," in *GLOBECOM*, November 1997.